EPL342 –Databases

Lab 11

SQL-DML IV (Indexes)

What is an Index?

- An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view
  - An index contains keys built from one or more columns in the table or view
  - These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently

- Simply put, an index is a pointer to data in a table
  - Very similar to an index at the end of a book
    - E.g. if you want to reference all the pages in a book that discuss a certain topic, you first refer to the index, which lists all topics alphabetically and are then referred to one or more specific page numbers

Why use an Index?

- An index helps speed up SELECT queries and WHERE clauses
- but it slows down data input, with UPDATE and INSERT statements
- Indexes can be created or dropped with no effect on the data

Creating Indexes

- Creating an index involves the CREATE INDEX statement
  - allows you to name the index
  - to specify the table and which column or columns to index,
  - indicate whether the index is in ascending or descending order.
- Indexes can also be unique, similar to the UNIQUE constraint
  - the index prevents duplicate entries in the column or combination of columns on which there's an index

CREATE INDEX Command

- **Single-Column Indexes**

CREATE INDEX index_name ON table_name (column_name)

- **Unique indexes**
  - Unique indexes are used not only for performance, but also for data integrity.
  - A unique index does not allow any duplicate values to be inserted into the table

CREATE UNIQUE INDEX index_name on table_name (column_name)

- **Composite Indexes**
  - A composite index is an index on two or more columns of a table. Following is the basic syntax.

CREATE INDEX index_name on table_name (column1, column2)

- **Implicit Indexes**
  - Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

DROP INDEX Command

- An index can be dropped using MS SQL SERVER DROP command.
  - Care should be taken when dropping an index because performance may be slowed or improved.

DROP INDEX tablename.index_name

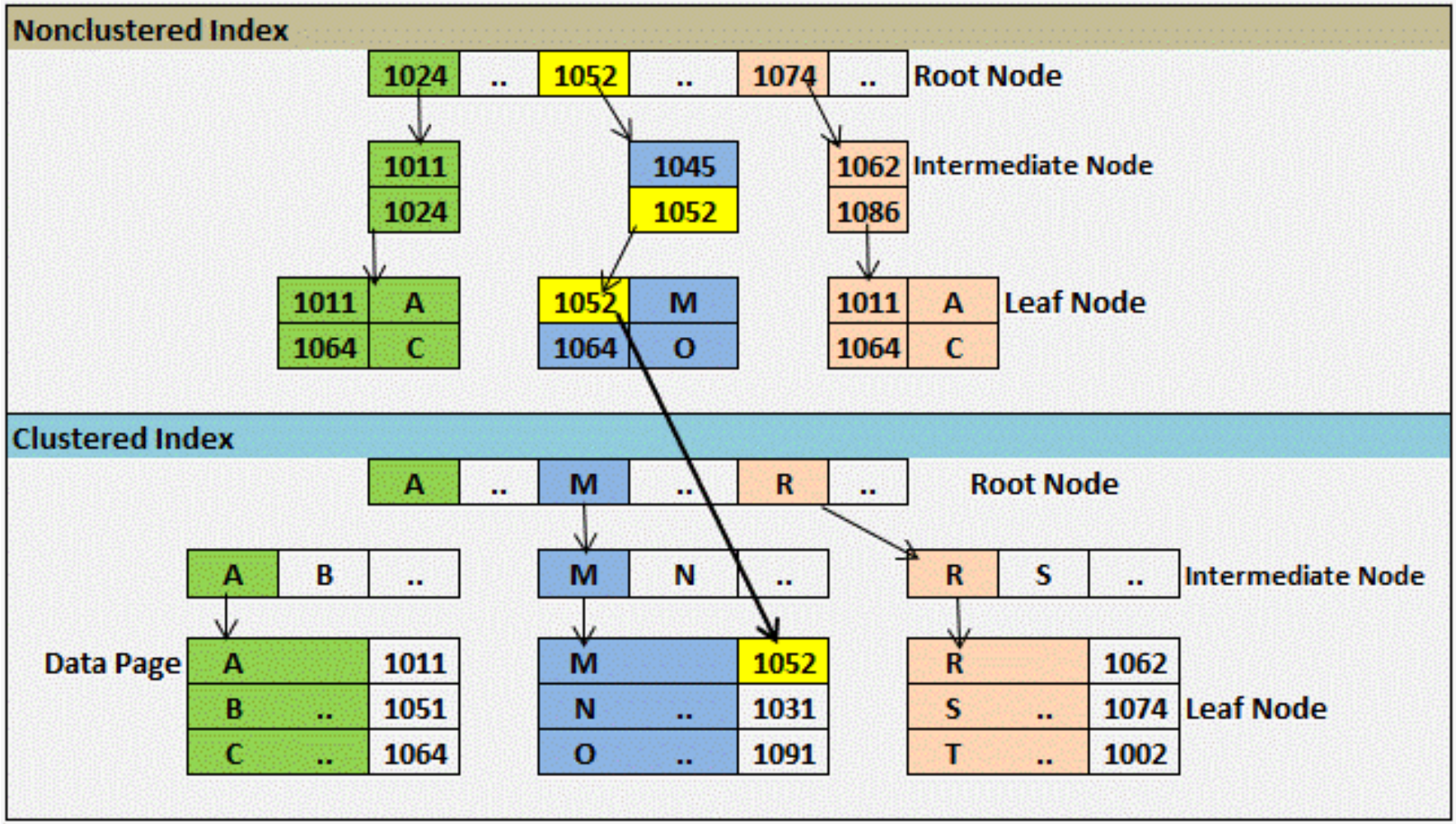
Clustered vs Non-clustered Index

- SQL Server has two types of indexes: clustered index and non-clustered index
- Clustered
  - Clustered indexes sort and store the data rows in the table or view based on their key values
  - These are the columns included in the index definition
  - There can be only one clustered index per table, because the data rows themselves can be stored in only one order
  - The only time the data rows in a table are stored in sorted order is when the table contains a clustered index
  - When a table has a clustered index, the table is called a clustered table
  - If a table has no clustered index, its data rows are stored in an unordered structure called a heap

Clustered vs Non-clustered Index

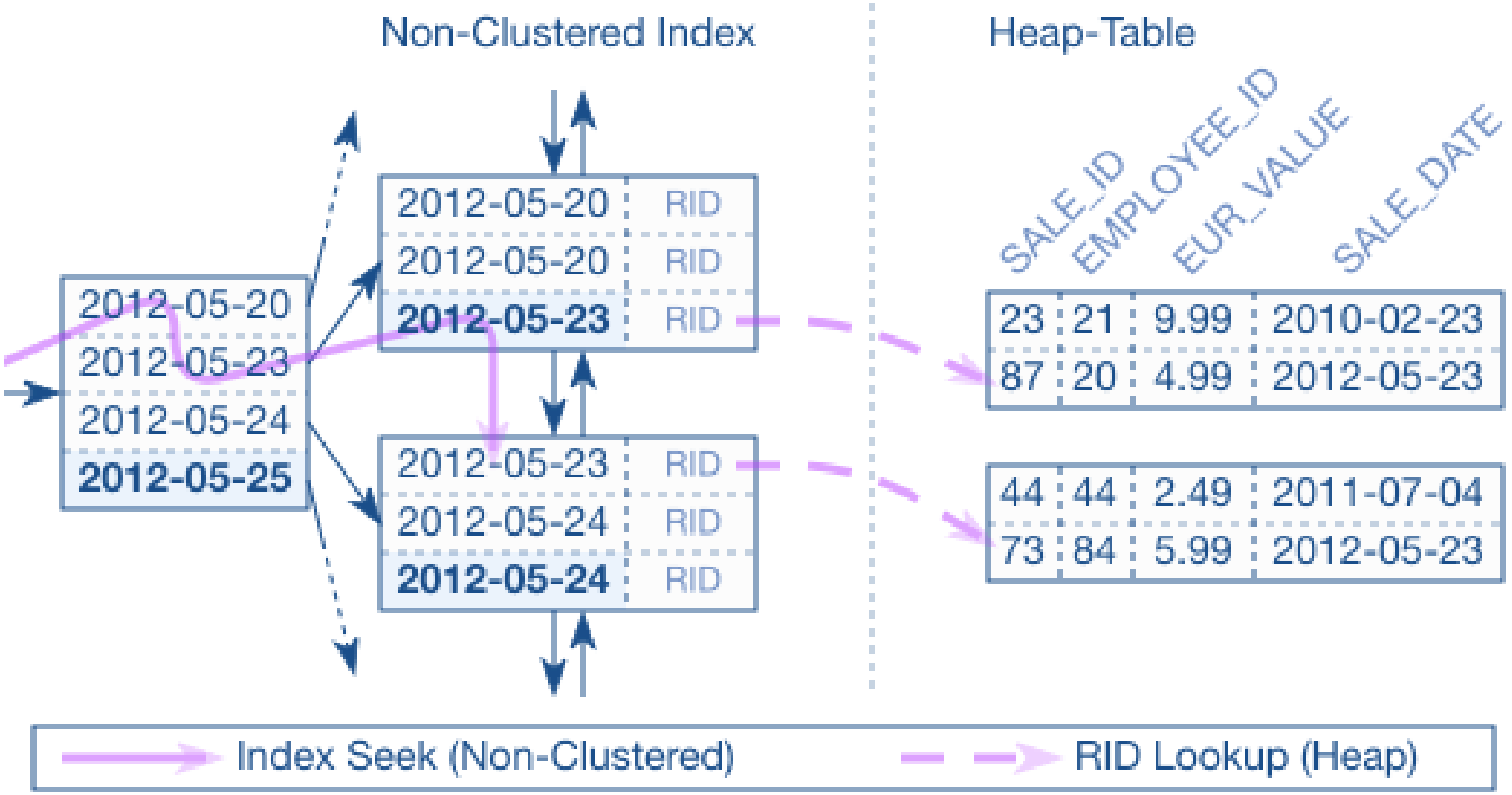
- Nonclustered
  - Nonclustered indexes have a structure separate from the data rows
  - A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value
  - The pointer from an index row in a nonclustered index to a data row is called a row locator
    - The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table
    - For a heap, a row locator is a pointer to the row
    - For a clustered table, the row locator is the clustered index key.
  - You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, and execute fully covered, indexed, queries

Clustered vs Non-clustered Index

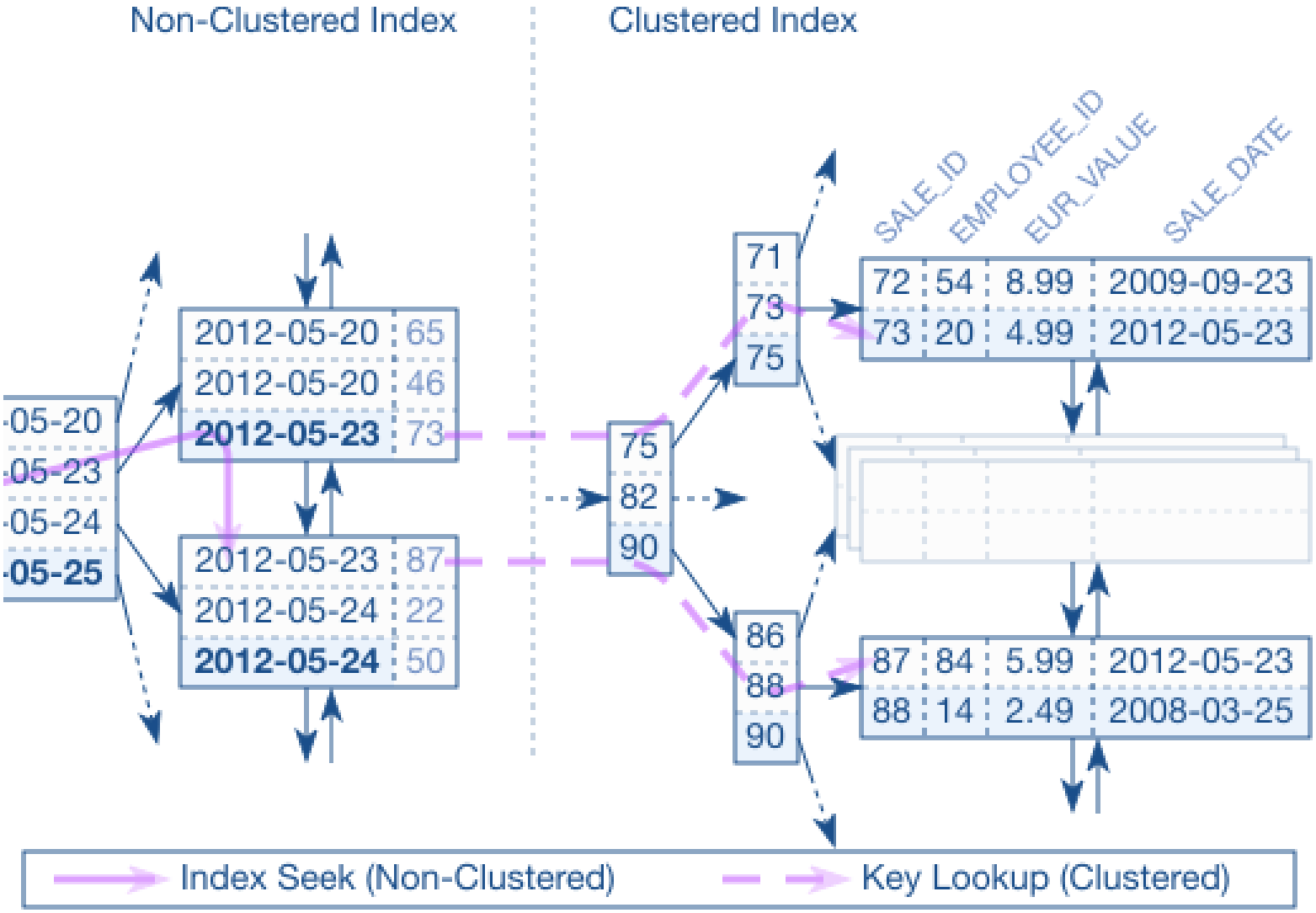


Nonclustered Index structure navigation on Clustered Index Structure in SQL Server

Clustered vs Non-clustered Index



Clustered vs Non-clustered Index



When to Avoid Indexes?

- Although indexes are intended to enhance the performance of databases, there are times when they should be avoided.
  - Indexes should not be used on small tables
  - Tables that have frequent, large batch update or insert operations should not be indexed
  - Indexes should not be used on columns that contain a high number of NULL values
  - Columns that are frequently manipulated should not be indexed

Practice

- Find and suggest indexes for the Northwind example database based on the queries of the previous labs
  - Would you use clustered on non-clustered indexes? Why?